УДК 004

Разработка алгоритма машинного обучения

Толмашов Олег Петрович
Сибирский федеральный университет
студент

Михайлов Андрей Сергеевич
Сибирский федеральный университет
студент

Аннотация
Пример работы алгоритма линейной регрессии для предсказания цены дома, основываясь на его площади и количестве этажей.
Ключевые слова: машинное обучение, линейная регрессия.

Development of machine learning algorithm

Tolmashov Oleg Petrovich
Siberian Federal University
student

Mikhailov Andrey Sergeevich
Siberian Federal University
student

Abstract
An example of a linear regression algorithm for prediction house prices based on house’s area and amount of rooms.
Keywords: machine learning, linear regression.

Машинное обучение — это область искусственного интеллекта, изучающая алгоритмы, способных обучаться. Разделяют два вида обучения: с учителем, и без него. В данной статье будет рассмотрено обучение с учителем, в частности модель линейной регрессии. Для обучения будет использоваться метод градиентного спуска. В процессе обучения качество модели будет определяться с помощью функции стоимости, для итоговой валидации будет использовано среднее процентное отклонение. В качестве обучающей выборки будет использован набор данных, содержащий 87 примеров стоимости домов, площади этих домов в квадратных метрах, а также количества комнат.

Модель представляет собой уравнение первой степени (рис.1).

$$h(x_1, x_2) = \theta_0 + \theta_1 * x_1 + \theta_2 * x_2$$

Рисунок 1 – Гипотеза, $\theta_0, \theta_1, \theta_2$ - параметры, найти которые необходимо в процессе обучения

Далее необходимо определить такое понятие, как функция стоимости, которую можно найти по формуле (рис.2).

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

Рисунок 2 – Формула функции стоимости

Главной целью при обучении модели будет минимизация функции стоимости. Чем меньше значение функции стоимости, тем более точный прогноз сможет дать модель. Для обучения будет использован метод градиентного спуска, который работает следующим образом: каждый параметр θ в цикле рассчитывается по следующей формуле (рис.3).

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

Рисунок 3 – Формула расчета параметров

Вычисление параметров происходит до тех пор, пока график функции стоимости не достигнет своего абсолютного минимума.

Для реализации этого алгоритма на языке python будут использованы библиотеки pandas, numpy и sklearn. Pandas - библиотека для работы с данными и анализом этих данных. Numpy - библиотека для работы с массивами и линейной алгеброй. Sklearn - библиотека для работы с алгоритмами машинного обучения на языке python. В данном случае из библиотеки sklearn будет использован только метод для разделения данных на тестовый и тренировочные наборы данных. Также будет использован алгоритм нормализации (mean normalization), так как изначальные данные имеют большие значения и это может негативно отразиться на обучении.

Следующий алгоритм позволяет вычислить параметры для модели (рис.4).

```python
from __future__ import division
import pandas as pd
import numpy as np
from sklearn import model_selection

# upload data
raw_data = pd.read_csv('training_data2.txt',names=['feet', 'rooms', 'price']).astype('float64')

# mean normalization
max_feet = raw_data['feet'].max()
min_feet = raw_data['feet'].min()
avg_feet = raw_data['feet'].mean()
max_rooms = raw_data['rooms'].max()
min_rooms = raw_data['rooms'].min()
avg_rooms = raw_data['rooms'].mean()
for row in raw_data.values:
    row[0] = (row[0] - avg_feet) / (max_feet - min_feet)
    row[1] = (row[1] - avg_rooms) / (max_rooms - min_rooms)

# split all data to train and test sets
train_data, test_data = model_selection.train_test_split(raw_data, test_size=0.3, train_size=0.7, shuffle=False)

# parameters
thetas = [0, 0, 0]
# learning rate
learning_rate = 0.01
# amount of iterations
iterations = 20000

# hypothesis
def h(x1, x2):
    return thetas[0] + thetas[1] * x1 + thetas[2] * x2

def cost_function(data):
    return sum([(h(row[0], row[1]) - row[-1])**2 for row in data.values]) / len(data.values)

cost = cost_function(train_data)
print('First cost = {}'.format(cost))

print('Started learning...')
for i in range(iterations):
    new_thetas = [0, 0, 0]
    errors = [h(row[0], row[1]) - row[-1] for row in train_data.values]
    new_thetas[0] = thetas[0] - learning_rate * (1/len(train_data.values)) * sum(errors)
    errors = [(h(row[0], row[1]) - row[-1]) * row[0] for row in train_data.values]
    new_thetas[1] = thetas[1] - learning_rate * (1/len(train_data.values)) * sum(errors)
    errors = [(h(row[0], row[1]) - row[-1]) * row[1] for row in train_data.values]
    new_thetas[2] = thetas[2] - learning_rate * (1/len(train_data.values)) * sum(errors)
    thetas = new_thetas

print('Finished learning!')
print('New cost = {}'.format(cost_function(train_data)))
print('Calculated parameters: {}'.format(thetas))
```

Рисунок 4 – Исходный код алгоритма

Результат работы вышеприведенного алгоритма на рис.5.

```
First cost = 138063566937.71875
Started learning...
Finished learning!
New cost = 4451140051.0755005
Calculated parameters: [347467.5399106569, 505119.2685525412, 45858.70943148974]
```

Рисунок 5 – Результат работы алгоритма

Далее необходимо провести валидацию модели. Валидация будет проводиться при помощи среднего процентного отклонения (рис.6).

```
def mean_absolute_percentage_error(data):
    deviations = []
    for row in data.values:
        predicted = h(row[0], row[1])
        percentage_deviation = abs((row[-1] - predicted) / row[-1]) * 100
        deviations.append(percentage_deviation)
    mape = sum(deviations) / len(deviations)
    return mape

train_mape = mean_absolute_percentage_error(train_data)
print('Mean absolute percentage deviation for training set is {:.2f}%'.format(train_mape))
test_mape = mean_absolute_percentage_error(test_data)
print('Mean absolute percentage deviation for test set is {:.2f}%'.format(test_mape))
```

Рисунок 6 – Валидация модели

Результат работы вышеприведенного алгоритма (рис.7).

```
Mean absolute percentage deviation for training set is 16.82%
Mean absolute percentage deviation for test set is 15.46%
```

Рисунок 7 – Результат валидации модели

Из результата становится понятно, что среднее процентное отклонение для данных, на которых производилось обучения, равно 16,82%, а среднее процентное отклонение для данных, которые не были использованы в обучении, равно 15,46%. Следовательно, данный алгоритм может быть использован для прогнозирования примерной цены дома, основываясь на его площади и количестве комнат.

Таким образом, данный результат показывает, что алгоритмы машинного обучения, в частности линейной регрессии, могут упростить жизнь человека в разных сферах его деятельности. Например, подобный алгоритм линейной регрессии может быть использован для прогнозирования цены на аренду жилья в зависимости от времени года или цены на подержанные автомобили, основываясь на их состоянии и годе выпуска.

Библиографический список

1. Линейная регрессия URL: http://www.machinelearning.ru/wiki/index.php?title=Линейная_регрессия (дата обращения 11.09.2018)
2. What machine learning is and why it matters URL: https://www.sas.com/en_us/insights/analytics/machine-learning.html (дата обращения 12.09.2018)
3. Course of machine learning URL: https://www.coursera.org/learn/machine-learning (дата обращения 03.09.2018)